WEST Search History

DATE: Monday, June 02, 2003

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
DB=EPAB,DWPI; PLUR=YES; OP=ADJ			
L7	L6 and (rpc or (remot$ adj procedur$ adj call$))	3	L7
L6	(rebuild$ or reconstruct$) near4 object	600	L6
DB=USPT; PLUR=YES; OP=ADJ			
L5	l1 same (rpc or (remot$ adj procedur$ adj call$))	7	L5
L4	l1 near12 rpc	1	L4
L3	L2 and l1	10	L3
L2	((709/330)!.CCLS.)	174	L2
L1	(rebuild$ or reconstruct$) near4 object	3048	L1

END OF SEARCH HISTORY

WEST Search History

DATE: Monday, June 02, 2003

Set Name side by side	**Query**	**Hit Count**	**Set Name** result set
DB=USPT; PLUR=YES; OP=ADJ			
L5	l1 same (rpc or (remot$ adj procedur$ adj call$))	7	L5
L4	l1 near12 rpc	1	L4
L3	L2 and l1	10	L3
L2	((709/330)!.CCLS.)	174	L2
L1	(rebuild$ or reconstruct$) near4 object	3048	L1

END OF SEARCH HISTORY

US005764915A

United States Patent [19]

Heimsoth et al.

[11] Patent Number: 5,764,915

[45] Date of Patent: Jun. 9, 1998

[54] OBJECT-ORIENTED COMMUNICATION INTERFACE FOR NETWORK PROTOCOL ACCESS USING THE SELECTED NEWLY CREATED PROTOCOL INTERFACE OBJECT AND NEWLY CREATED PROTOCOL LAYER OBJECTS IN THE PROTOCOL STACK

[75] Inventors: **Daniel Dean Heimsoth**, Cary, N.C.; **Gary Randall Horn**, O'Fallon, Mo.; **Mohan Sharma**, Austin, Tex.; **Laurie Beth Turner**, Atlanta, Ga.; **Leo Yue Tak Yeung**, Austin, Tex.

[73] Assignee: **International Business Machines Corporation**, Armonk, N.Y.

[21] Appl. No.: **611,849**

[22] Filed: **Mar. 8, 1996**

[51] **Int. Cl.**^{6} **G06F 13/14**; G06F 15/173

[52] **U.S. Cl.** **395/200.57**; 395/200.58; 395/200.59; 395/200.6

[58] **Field of Search** 370/258, 389; 395/200.01, 200.09, 700, 200.57, 200.58, 200.59, 200.6; 390/200.6

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,307,490	4/1994	Davidson et al.	395/684
5,421,016	5/1995	Conner et al.	385/700
5,548,723	8/1996	Pettus	395/200.01
5,548,726	8/1996	Pettus	395/200.09
5,590,124	12/1996	Robins	370/258
5,640,394	6/1997	Schrier et al.	370/389

FOREIGN PATENT DOCUMENTS

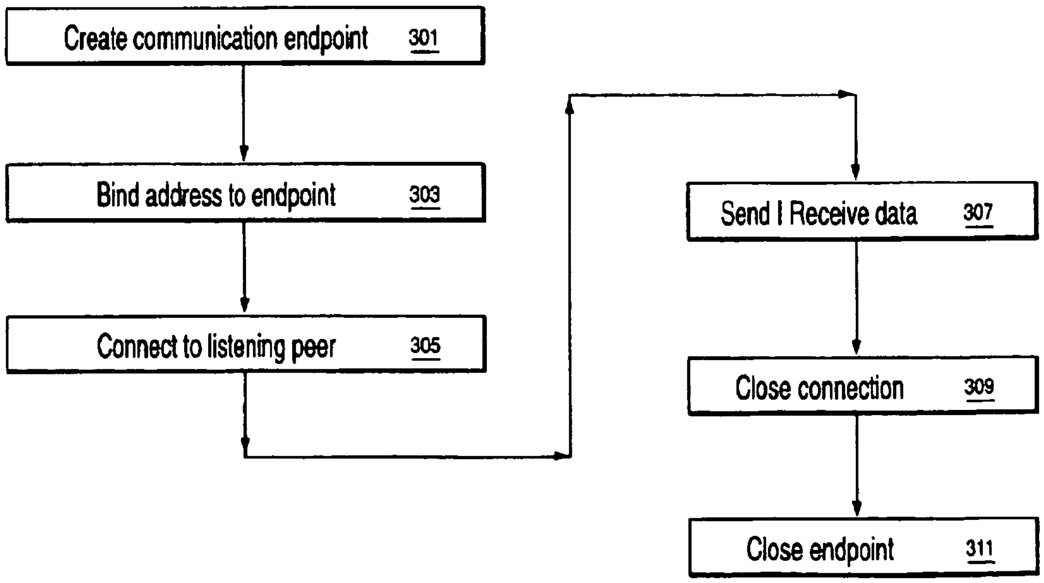
WO95/17060	6/1995	WIPO .
WO95/17065	6/1995	WIPO .
WO95/17066	6/1995	WIPO .
WO95/17720	6/1995	WIPO .

Primary Examiner—Thomas C. Lee
Assistant Examiner—David Ton
Attorney, Agent, or Firm—Jeffrey S. Labaw

[57] **ABSTRACT**

An object oriented protocol interface for establishing a communication path between communication endpoints in a computer network. The generic nature of the interface allows any and several protocol layers to be developed from the same set of protocol class objects. The interface to a communication endpoint for a client application is defined by instantiating a network definition object for the communication endpoint from a network definition class object. The communication endpoint itself is represented by instantiating a network address object from a network address class object. The protocol layers which form the protocol stack are derived from a set of protocol interface objects from a protocol interface class object and a set of protocol layer objects from a protocol layer class object. The objects forming each of the layers in the protocol stack differ in their capabilities according to their respective layer and the protocol which is provided by the protocol stack. The communication path is established in the protocol stack by calling methods in the sets of protocol interface and protocol layer objects.

20 Claims, 20 Drawing Sheets


Create communication endpoint 301
Bind address to endpoint 303
Connect to listening peer 305
Send / Receive data 307
Close connection 309
Close endpoint 311

End of Result Set

Generate Collection Print

L5: Entry 7 of 7 File: USPT Jun 9, 1998

DOCUMENT-IDENTIFIER: US 5764915 A
**** See image for Certificate of Correction ****
TITLE: Object-oriented communication interface for network protocol access using the selected newly created protocol interface object and newly created protocol layer objects in the protocol stack

Detailed Description Text (182):
For every communication endpoint which a user creates, there must be a unique ID that must be maintained to associate an endpoint to a stack of protocol layer objects. The protocol layer objects represent the endpoint on the server process. Note that this correspondence is one-to-one. For this purpose, during the creation of the endpoint using the TACcessDefinition:: Instantiate() method, the TAccessOp object is created. The TAccessOp object then creates a ClientStackHead object which represents the client side of communication. The AccessOp object is then flattened and sent to the server using either the RPC or IPC mechanism by the ClientStackHead. The server then rebuilds the TAccessOp object from the data stream using stream-in operator of TNetworkOperation and calls the TAccessop::Execute() method. This function creates a ServerStackHead object which creates the protocol layer objects from the protocol interface objects and keeps a list of the pointers to these protocol layer objects in the TServerStackHead object. The TServerStackHead pointer is stored in a global table of the network server and the index is streamed-out to the client. The TClientStackHead object stores the ServerStackHead ID and uses it for all subsequent operations. Thus, the ServerStackHeadID serves as a unique ID between a client and the server. Subsequent requests such as a TBindOp when received by a server, it locates the corresponding server stack head using the ID that is passed in the TBindOp.

Detailed Description Text (185):
1. ProcessOperation: This function is called by the TProtocolInterface or TAccessDefinition objects for the TClientStackHead to process the TNetworkOperation object. This function flattens the TNetworkOperation object and send the flattened operation object to the server using RPC/IPC mechanisms provided by the system. This function also rebuilds the NetworkOperation object when the server responds to the request that was sent. Basically, this function sends and receives NetworkOperation objects to and from the server.

Detailed Description Text (196):
FIG. 9D illustrates the flow of requests from client to server and vice-versa. As explained above, an endpoint is represented by the TAccessDefinition object on the client and a stack of TProtocolLayer objects on the server. The communication between the client and the server is managed by the TclientStackHead and the TServerStackHead objects. The figure shows two endpoints 725. A network request from the client is sent to the ServerProcess/Thread 727 as a TNetworkOperation object using the system RPC or IPC mechanisms. The ServerProcess rebuilds the TNetworkoperation object, then locates the TServerStackHead object that represents the endpoint1, and routes the request to ServerStackHead 1 729. Similar processing is done to route the request from endpoint 2 to ServerStackHead2 that represents this endpoint on the server. The ServerStack1 then calls TNetworkOperation::Execute() method which then calls an appropriate method in TTransportLayer of stack1 733. The TTransportLayer may choose to call a method in the TNetworklayer which then sends the request to the TFamilyLayer 737. The family